The Turtle Prize - Lesson 1 - Sequences

Contents

Introduction

Students learn to program by creating graphics using Python's turtle library. After spending some time creating graphics, students can enter their results into the Turtle Prize to win a Raspberry Pi computer and an electronics experimenter's kit.

Students are encouraged to work in pairs, and if a pair wins, then both members will win their own prize.

Learning objectives

- Students will learn that programs are made up of a sequence of computer codes that have to be written exactly.
- Students will learn to write a Python program that draws a simple picture.
- Students will understand what a computer library is.

Resources

- 15 copies of lesson 1 handout

Lesson Summary

- An introduction to the Turtle Prize.
- Program your robot game.
- Draw your initial with the turtle library.

An introduction to the Turtle Prize : 5 minutes

Tell students about the Turtle Prize, that we're going to create some graphics using a computer. But we're not going to use a graphics package (that someone else wrote), we're going to write our own program!

The first lesson will be about learning some of the basics of computer programming, and learning how to draw the initial of your name with computer codes.

The second lesson will be spent creating entries for the Turtle Prize.

The winning entry will be judged on originality and how good it looks. The winner across all entrants and schools will win a Raspberry Pi and an electronics experimenter's kit.

Programming your robot : 25 minutes

The most fundamental part of programming a computer is understanding that programs are made up of a sequence of commands.

It's important for us to also understand that a computer doesn't have any intelligence - it has to be told exactly what to do. So when the program is wrong, the computer does the wrong thing.

Ask the students to design a (simple) obstacle course by moving chairs about. Have a start and end point.

Ask the students what commands the robot will need, and to write them down on their handouts. For example `forward(10)` will make the robot take 10 steps.

Explain that the 10 is called an argument. Some codes need arguments, others don't.

Most courses can be solved with the following:

- left(degrees),
- right(degrees),
- forward(steps).

And we always have a `start` and `stop` code with associated sound effects!

Ask the students to write down a program in their handouts that will navigate a robot successfully through the obstacle course.

Now ask for a volunteer robot and a volunteer programmer to see how well they complete the course. Force the robot to follow the programmer's instructions exactly.

Draw your initial : 25 minutes

Explain that we'll now write a sequence of codes using a computer programming language called Python. The codes will make a simple picture.

Talk through the first turtle program. The numbers on the left are line numbers in the file and match those in the handout.

```
1 from turtle import *
2 forward(100)
3 done()
```

1. imports the turtle library. A library is a set of codes that we can borrow for our program. In this case it makes it easy for us to draw pictures.
2. move the turtle forward by 100 pixels. The 100 is the argument to the forward code.
3. finishes the program.

Get the students to work in pairs on a computer. Show the students how to start Idle, and explain that this is the program they'll use to write the codes.

In Idle, create a new program by clicking `file->new`.

Then ask the students to copy out the first turtle program from the handout into this new window.

When everyone has finished, they need to save their files. Ask them to save their file as a combination of their names with a `.py` at the end.

Saving the file as `turtle.py` will break the program! If this happens, you have to locate 2 files; `turtle.py` and `turtle.pyc` and delete them from the computer.

So for example, Alice and Bob would save as `alicebob.py`

After all students have saved, ask them to press `f5` to run the program. If they've correctly typed the program they should have a window popped up with a horizontal line in it. Check everyone's screen to see this has happened.

Explain that every time they make a change they should test it by running it. They can do this by `f5`, if the file has changed they'll get a box asking if they want to save. Encourage students to get into the habit of pressing `f5` and then `enter` to keep the time spent doing this as short as possible.

Now ask the students to take it in turns to create their initials using the *useful turtle commands* in the handout. If they haven't finished, set it as homework.

If students get stuck ask them to draw their initial on paper first.

Encourage students to take turns coding. One writes code and the other helps to come up with ideas. Also encourage them to look at other pair's graphics and the code that made them.

The done() command

So that you know, the `done()` command finished the program "nicely" and allows the exit button on the window to work. If students miss it out, the program will still work but they'll have problems closing the window.

Extension activity

After creating their initials, ask students to draw a picture, or to explore the other turtle commands detailed in the handout.

Plenary : 5 minutes

Ask how a computer runs a program. Check that the students understand that computers have to follow specific instructions in a sequence. If any instructions are badly written the computer will complain with an error.

Ask what a computer library is. Check that students understand a library is a set of codes written that we can borrow instead of having to write our own. We are using the turtle library in this lesson to make it easy to draw pictures.

Homework

Research to be carried out using the internet.

- Why is Python called Python?
- Who is Idle named after?
- Use an image search engine to find some inspirational pictures of graphics made with the turtle library. Try and understand the code used.

Outcome

All students:

- Understand that computer programs are written in a sequence of instructions.
- Computers are stupid, and all instructions have to be written correctly.
- Have used Python to create some simple graphics.

Most students:

- Have used the turtle library to create their initial.

Some students:

- Have further experimented with the other turtle commands.